A Probabilistic Method for Cooperative Hierarchical Aggregation of Data in VANETs

Christian Lochert, Björn Scheuermann, and Martin Mauve

Abstract

We propose an algorithm for the hierarchical aggregation of observations in dissemination-based, distributed traffic information systems. Instead of transmitting observed parameters directly, we propose soft-state sketches—an extension of Flajolet-Martin sketches—as a probabilistic approximation. This data representation is duplicate insensitive, a trait that overcomes two central problems of existing aggregation schemes for VANET applications. First, when multiple aggregates of observations for the same area are available, it is possible to combine them into an aggregate containing all information from the original aggregates. This is fundamentally different from existing approaches where typically one of the aggregates is selected for further use while the rest is discarded. Second, any observation or aggregate can be included into higher level aggregates, regardless if it has already been previously—directly or indirectly—added. Those characteristics result in a very flexible aggregate construction and a high quality of the aggregates. We demonstrate these traits of our approach by a simulation study.

Key words: VANET, Aggregation, Sketch, Information Dissemination, Arithmetic Coding

1. Introduction

Cooperative information gathering and sharing forms a prominent class of VANET applications. For instance, approaches that disseminate traffic information like TrafficView [1] or SOTIS [2, 3] as well as a system to exchange information on free parking places [4] have been designed. These applications have in common that they distribute measurement results obtained by the participating cars in a comparatively large area.

Typically, this is accomplished in the way schematically shown in Figure 1. Each car makes observations. An observation is essentially some measured value (traffic density, free parking places, road condition, ...), related to a position in space (typically a road segment or a small area) and a point in time when the observation has been made. All or part of the locally stored information is periodically single-hop broadcasted in beacon packets. Upon reception of such a beacon, a node incorporates the received data into the local knowledge base. By comparing the timestamps of observations, it can ensure that always the most up-to-date value for each position is stored and redistributed. However, if we assume that the spatial density of points for which observations are made is approximately constant, the amount of data increases quadratically

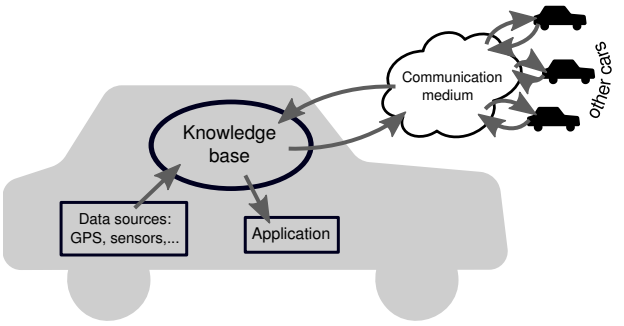


Figure 1: System model of a typical VANET data dissemination application.

with the covered radius. Thus, the amount of data to be broadcasted by each car will likewise increase quickly. This is fatal for the scalability of such a system [5].

To overcome this problem, the use of hierarchical data aggregation has been proposed: with increasing distance, observations concerning larger and larger areas (or road segment lengths) are combined into one single value. Such an aggregated value could, for example, be the average speed on a longer road segment [1, 2, 3], or the percentage of free parking places in a part of a city [4]. Coarse aggregates are made available at greater distances, more detailed data is kept only in the near vicinity. However, even though the idea has often been mentioned [1, 2, 3, 4, 6, 7, 8], central problems of such an approach have, so far, remained unsolved.

A fundamental issue that arises is that aggregates cannot, like single observations, be directly compared with respect to the up-to-dateness and completeness of the contained data. They are created by cars that will typically not have the most up-to-date measurements for all underlying points available. Therefore, multiple aggregates for the same area may exist, based on different, but likely overlapping knowledge. To decide which one is based on "better" underlying data is hard, if not impossible. While this problem has been mentioned before in the literature (see, for example, [6, 1]), no fundamental solution has so far been proposed.

In this paper—a revised and extended version of [9]—we propose an algorithm that solves this issue. To the best of our knowledge, this is the first approach that can handle aggregates of overlapping entries. We achieve this by a special data representation: both single observations and aggregates in our scheme do not carry the value of, e. g., the number of free parking places directly, but instead contain an approximation of it in form of a so-called soft-state sketch. The herein introduced soft-state sketches are a data structure based on modified Flajolet-Martin sketches [10].

While soft-state sketches do still not provide a way to compare the quality of two aggregates directly, they allow for something even better: in our scheme, multiple aggregates for the same area can be merged, yielding a new one that incorporates all the information contained in any one of the aggregates. This is fundamentally different from all existing dissemination approaches, where two aggregates describing the same

area cannot be merged[1]. In our scheme there is no need to decide which aggregate contained more up-to-date information since the resulting aggregate comprises all the information from all aggregates that have been merged.

Our approach also allows observations or lower-level aggregates to be integrated into an already existing higher-level aggregate at any time. This, too, is not possible with any previously existing hiearchical aggregation approach, because it cannot be determined which data is already present in the aggregate and interesting aggregates like sums or averages are typically duplicate sensitive.

Apart from making decisions regarding the aggregate quality unnecessary, the proposed scheme also largely eases the generation of good aggregates. A node would usually have to collect data on a significant fraction of the covered area before an aggregate that likely constitutes a good representation can be formed. With our scheme, the aggregate can instead be maintained while being passed around in the network, always incorporating new information on-the-fly.

The remainder of this paper is structured as follows. In the next section we review aggregation as it has been proposed for VANET applications, as well as some previous uses of Flajolet-Martin sketches in the networking area. Thereafter, we quickly recapitulate Flajolet-Martin sketches in Section 3, to set the stage for the introduction of soft-state sketches and the detailed discussion of our algorithm in Section 4. We subsequently propose two extensions in Section 5. In Section 6, we present and discuss the results of a simulation-based evaluation of the algorithm in a VANET city scenario. Finally, we conclude this paper with a summary in Section 7.

2. Related Work

Recently, many convenience applications for VANETs have been discussed, and often they use some form of data dissemination. In the Self-Organizing Traffic Information System (SOTIS) [2, 3], information on the traffic situation is distributed opportunistically, by sending periodic beacons containing the knowledge of the sending node on the traffic situation in a larger surrounding. The authors also outline a (non-hierarchical) aggregation scheme, combining all the known information on each fixed-length road segment to one average value. Upon reception, a node considers an aggregate to be "better" if it has a newer timestamp. But since these timestamps are assigned when the aggregate is computed, this system exhibits the problems outlined in the introduction: a newly computed aggregate with a new timestamp is not necessarily based on the most up-to-date information, and aggregates representing largely disjoint knowledge can neither be identified as such nor can they be merged. For these reasons, a system like SOTIS could largely benefit from the aggregation scheme introduced here.

TrafficView [1] is another system for disseminating traffic information, similar to SOTIS in both aims and mechanisms. The authors of TrafficView also introduce a

[1] Any hierarchical aggregation scheme will provide a way to merge lower level aggregates into a higher level aggregate. This is not what we refer to. The point here is that two aggregates describing the same area are merged.

data aggregation scheme. Different from SOTIS, TrafficView distributes information on position and speed of single vehicles. The aggregation mechanism combines a number of "similar" vehicles in an adaptive way, aiming to minimize the introduced errors. Again, each aggregate is assigned a timestamp: TrafficView uses the minimum information generation time of the combined measurements. Consequently, similar problems as for SOTIS arise. A definite decision about the relative up-to-dateness of stored and received values can not reliably be made. Again, this could be overcome by using our probabilistic aggregation scheme.

A different application of data dissemination with aggregation in VANETs is proposed in [4]: the distribution of information on free parking places. The information on the current occupancy level is generated by local infrastructure at the parking lots, and, like in the traffic information systems above, disseminated by periodic beaconing. Aggregation is performed hierarchically in a quad-tree structure over the city area. Where SOTIS uses the time of generation of the aggregate and TrafficView the minimum of the generation times of the contained atoms, the parking guidance system compares aggregates by the *average* time of information generation of the atomic values contained in the aggregate. However, this again does not allow to decide reliably which of two aggregates is based on more precise data. Furthermore, it does not solve the duplicate problem, and hence does not allow merging aggregates for the same region. In this case our aggregation scheme could be applied as well, and would help to overcome these difficulties.

In [7, 8] Ibrahim et al. present CASCADE, an information dissemination protocol similar to SOTIS and TrafficView. They assume that vehicles form clusters on a highway. Detailed speed and position information is exchanged within the cluster. For a larger scope, "cluster records" contain aggregated information, in the form of relative positions of all cars to a cluster head and an average speed. Multiple cluster records form periodically broadcasted information messages. A strict dropping policy and life cycle are applied for removing old data. CASCADE reduces the amount of data transmitted about a cluster of cars, but it does not include mechanisms for merging aggregates.

Brønsted et al. propose protocols for disseminating information on the current road conditions [6]. Zone Flooding floods information within a predefined area, Zone Diffusion uses aggregation techniques. A road is divided into cells, multiple cells form an "environment representation". Nodes broadcast these representations periodically. The authors mention the problem of merging overlapping aggregates, but they do not detail how it can actually be solved.

A substantial amount of work has been put into optimizing the beaconing in dissemination applications. This includes adaptive beaconing intervals, aiming to give cars with more or newer knowledge preferred access to the medium. It also encompasses the selection of the optimal subset of the knowledge base to be incorporated into a transmission of limited size. Work in this area can, for example, be found in [11, 12, 13, 14, 15].

When speaking about applications for VANETs in a real environment, questions about security arise. For instance, it has been studied how VANETs and aggregated messages in them can be made tamper-proof or resistant to cheaters [16, 17, 18, 19, 20, 21]. Since all these questions are largely orthogonal to the fundamental challenge

of how aggregates should be computed, compared, and merged, their ideas can be combined with our approach.

In [22], Flajolet-Martin sketches are used for robust in-network aggregation in sensor networks in the presence of packet loss or node failures. The coordinated collection of information towards a sink in the sensor network is considered, as opposed to continuously updated distribution of information to all nodes as in this paper. The problem of removing old information, which we solve here, does therefore not arise. In [23], the use of Flajolet-Martin sketches for spatio-temporal database indexes is discussed. While these techniques could potentially be used to speed up specific queries in a centralized traffic information system, the problem is rather different from what we consider here, and their approaches are not transferable to a distributed, dissemination-based system. Several other data structures for probabilistic, duplicate insensitive counting have been proposed lately. Examples are LogLog sketches [24] and HyperLogLog sketches [25]. We use Flajolet-Martin sketches as a basis for our soft-state sketches because they allow for a modification that enables the removal of old data from the aggregates.

3. Flajolet-Martin Sketches

As mentioned above, our aggregation mechanism is based on soft-state sketches. Soft-state sketches, in turn, are derived from Flajolet-Martin sketches. Therefore, before we turn towards our own contributions, we now summarize the relevant aspects of Flajolet-Martin sketches.

A Flajolet-Martin sketch (also called "FM sketch" or in the following simply "sketch") is a data structure for probabilistic counting of distinct elements that has been introduced in [10], and was originally intended to be used to speed up database queries. An FM sketch represents an approximation of a positive integer by a bit field $S = s_1, \ldots, s_w$ of length $w \geq 1$. The bit field is initialized to zero at all positions. To add an element x to the sketch, it is hashed by a hash function h. This hash function has geometrically distributed positive integer output; the probability that the hash value $h(x)$ of x is equal to i is given by

$$P(h(x) = i) = 2^{-i}. \tag{1}$$

The entry $s_{h(x)}$ is then set to one. (With probability 2^{-w} we have $h(x) > w$; in this case, no operation is performed.) A hash function with the necessary properties can easily be derived from a common hash function with uniformly distributed bit string output by using the position of the first 1-bit in the output string as the hash value.

The central result of [10] is that an approximation $C(S)$ of the number of distinct elements added to the sketch can be obtained from the length of the initial, uninterrupted sequence of ones, given by

$$Z(S) := \min(\{i \in \mathbb{N}_0 \mid i < w \wedge s_{i+1} = 0\} \cup \{w\}) \tag{2}$$

by calculating

$$C(S) := \frac{2^{Z(S)}}{\varphi}, \tag{3}$$

with $\varphi \approx 0.77351$.

The variance of $Z(S)$ is quite significant, and thus the approximation is not very accurate. To overcome this, instead of only one sketch a set of sketches can be used to represent a single value, trading off accuracy against memory. The respective technique is called Probabilistic Counting with Stochastic Averaging (PCSA) in [10]. With PCSA, each added element is first mapped to one of the sketches by using a uniformly distributed hash function, and is then added there. If m sketches are used, denoted by $S_1, \ldots, S_m$, the estimate for the total number of distinct items added is then given by

$$C(S_1, \ldots, S_m) := m \cdot \frac{2^{\sum_{i=1}^{m} Z(S_i)/m}}{\varphi}. \tag{4}$$

But, as [10] also states, this formula is rather inaccurate as long as the number of elements is below approximately $10 \cdot m$. As demonstrated in [26], the slighly modified formula

$$C(S_1, \ldots, S_m) := m \cdot \frac{2^{\sum_{i=1}^{m} Z(S_i)/m} - 2^{-\kappa \cdot \sum_{i=1}^{m} Z(S_i)/m}}{\varphi}, \tag{5}$$

with $\kappa \approx 1.75$ alleviates these inaccuracies.

PCSA with m sketches yields a standard error of approximately $0.78/\sqrt{m}$ [10, 27]. For many VANET applications, sufficiently good approximations are possible at reasonable sizes.

4. Sketch-based Aggregation in VANETs

FM sketches can be merged to obtain the total number of distinct elements added to any of them by a simple bit-wise OR. Important here is that, by their construction, repeatedly combining the same sketches or adding already present elements again does not change the results, no matter how often or in which order these operations occur. Furthermore, sketches are space efficient, because the covered value range grows exponentially with the bit field size w.

These are highly desirable properties for distributed VANET applications with in-network data aggregation. We will therefore now describe how these benefits of FM sketches can—in principle—be carried over when they are applied in VANET dissemination applications. We will also see that the application of unmodified FM sketches alone is not sufficient, which then leads us to the idea of soft-state sketches.

4.1. Creating and merging sketches

For the purpose of discussion in this section, let us consider a specific application: assume that we are interested in disseminating the number of free parking places. This will help to clarify the basic ideas for a concrete application; the same concepts and algorithms can be applied if other values are disseminated.

For the moment, we do not care about the measured quantities changing over time. As a first step, we use a sketch (or, with PCSA, a set of sketches) for each road segment. We assume that a car is able to observe the current number of free parking places while passing a road segment, e. g., by collecting data from sensors on the parking places, as

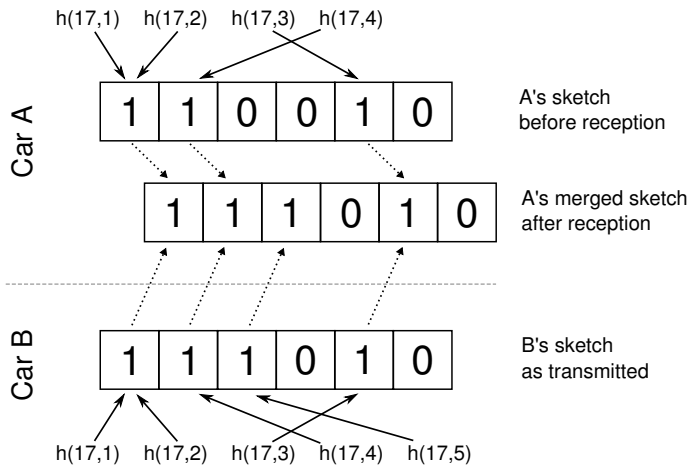


Figure 2: Generation and merging of FM sketch aggregates.

proposed in [28]. After passing a road segment with ID r and observing x free parking places on it, a car may add the tuples $(r,1),\ldots,(r,x)$ to the sketch for r, by hashing them and setting the respective bits. The same principle could be used if x was, for instance, the number of cars on the road segment as a measure for the traffic density.

The locally stored sketches for the road segments are periodically broadcasted to the node's one-hop neighbors. Upon reception, received and local sketches are merged by calculating the bit-wise OR. Figure 2 exemplifies this procedure. Two cars, A and B, make independent observations on the same road segment (with ID 17). A observes 4 free parking places and thus hashes the tuples $(17,1),\ldots,(17,4)$ into its sketch for road segment 17. B observes 5 free parking places, and consequently adds $(17,1),\ldots,(17,5)$. If A and B meet later, and A receives a transmission containing B's sketch, A merges them by bit-wise OR and obtains a new sketch, replacing its previous one. Obviously, this can be repeated whenever cars meet, such that information from further cars can be incorporated, exchanged, and passed on, thereby accumulating "knowledge" in the aggregate.

Note that the hashed tuples (r,i) are identical for different observers, the observed value determines only how many of them are added. If all observers use the same hash function (something that could easily be standardized), the same number of free parking places on the same road segment will set the same bits, a lower number will set subset thereof. Of course, in the current basic algorithm, bits that have once been set will never get unset again, and the sketch is therefore not able to follow decreasing values. We will therefore now discuss how to extend the data structure in order to overcome this limitation.

4.2. Soft-state sketches

The sketches will always represent the maximum of all ever observed values for each road segment; therefore, a method is needed to remove old observations. We accomplish that by modifying the original FM sketches. We use small counters of n

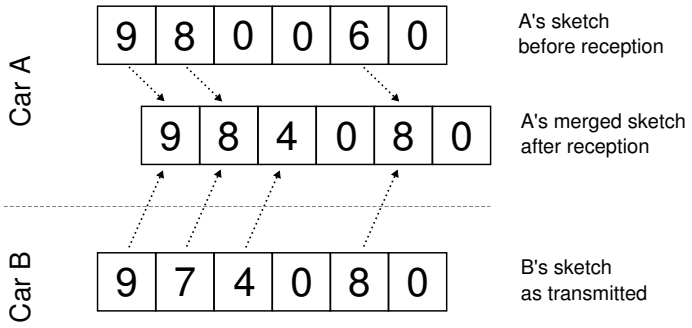


Figure 3: Merging of soft-state sketches.

bits length instead of single bits at each index position. These counters represent a time to live (TTL) in the range $0, \ldots, 2^n - 1$ for that bit. The operation of setting a bit to one after an observation is replaced by setting the corresponding counter to the maximum TTL, to $T := 2^n - 1$. Beacons containing the sketches are sent at regular intervals. Just before sending such a beacon with information from the local knowledge base, all counters in the locally maintained sketches are decremented by one, if they are not yet zero.

When incorporating a received sketch into the local knowledge base, the bit-wise OR is substituted by a position-wise maximum operation. This yields a soft-state variant of FM sketches, in which previously inserted elements essentially die out after their TTL has expired, unless they are refreshed by a newer observation. The merging is visualized in Figure 3. Car A receives an aggregate from car B, and updates its own soft-state sketch accordingly.

For obtaining the current value from a soft-state sketch, the algorithm remains essentially unchanged; still, the smallest index position with value zero is identified and used. Note that this incurs some delay if a bit position is no longer set in newer observations. Coming back to our above example of observing parking places, assume that no further observations are made which set a particular bit position (e. g., because there is no longer a free parking place being hashed to it). If the position had previously been set in an aggregate, then the TTL value will decrease over time until it arrives at zero.

As an extension, it is possible to consider the smallest index position with a value below some threshold instead of the smallest position being zero, by using

$$Z_t(S) := \min(\{i \in \mathbb{N}_0 \mid i < w \wedge s_{i+1} \leq T - t\} \cup \{w\}) \tag{6}$$

in the role of $Z(S)$—which equals $Z_T(S)$—in (3) or (5). The threshold t can be chosen arbitrarily and individually whenever evaluating a sketch, in the range between 1 and T. This yields an approximation $C_t(S)$ of the total distinct item count observed in the last t beaconing intervals. It therefore allows to dynamically choose a "cutoff horizon", thereby trading off between taking only more recent observations—that is, the most up-to-date data—into account and working on a larger data basis.

4.3. Hierarchical aggregation

Hierarchical aggregation is typically done on trees, often on symmetric and self-similar ones like quad-trees over the two-dimensional plane. But while it may for instance be expected that the traffic situation does not differ much among a set of similar and close-by road segments, or that the fraction of free parking places is relatively constant within a neighborhood, it might at the same time be vastly different not too far away: for example on the other side of a highway or a river. Therefore, a good aggregation scheme should respect the environment it is imposed on. We envision that such an aggregation hierarchy is pre-defined in the map data, following the underlying structure and grouping areas in a way that reflects their natural relations, like city districts or road hierarchies.

This is explicitly supported by our algorithm. Let L denote the set of locations for which observations can be made, like the entirety of road segments, or simply all points on the map. Many aggregates are possible—in principle, any arbitrary combination of locations could be aggregated. Possible aggregates are thus the (non-empty) elements of L's power set $\mathcal{P}(L)$. We may choose a subset $\mathcal{A}$ of these as the areas for which sketches are to be maintained:

$$\mathcal{A} \subseteq \mathcal{P}(L) \setminus \{\emptyset\} \tag{7}$$

The structure of $\mathcal{A}$ is not constrained by our method, though certain choices exhibit benefits.

Based on the outlined idea, hierarchical aggregation can be accomplished in the following way. We allot sketches for all elements in $\mathcal{A}$. Any observation made for some location $l \in L$ can immediately be incorporated into each aggregate for which the aggregated area A contains l. Consequently, in our example application of counting free parking places, these aggregates will contain the total counts in their respective areas.

It should be noted that not necessarily for each location a separate sketch needs to be maintained, so there is no necessity that $\forall l \in L : \{l\} \in \mathcal{A}$. Especially if L is large (or continuous!), it might make sense to maintain sketches only for areas encompassing multiple locations.

Information on small-scale areas will then typically be kept in the closer vicinity, while further away cars will preferably maintain and distribute larger-scale aggregates of the region. The duplicate insensitivity of the sketches allows for aggregates to be merged in just the same way as it has been introduced above for sketches of single locations. But in particular, any received sketch for some area A can immediately be incorporated into any *superordinate* aggregate A', where superordinate means that A is wholly covered by A', that is, $A \subseteq A'$.

To allow for incorporating received information into as many aggregates as possible, $\mathcal{A}$ will thus indeed often be a hierarchical tree structure, where for all $A_1, A_2 \in \mathcal{A}$ it holds that

$$A_1 \cap A_2 \neq \emptyset \quad \Longrightarrow \quad A_1 \subseteq A_2 \ \vee \ A_2 \subseteq A_1. \tag{8}$$

Even this, however, implies neither any symmetry nor the same depth of all subtrees of any given node. It also does not exclude cases where some higher-level area is not completely covered by smaller subareas. Therefore, this concept is much more powerful than the commonly used aggregation trees.

4.4. Practical issues and applicability

The simple parking place counting application as discussed above has a major drawback. In case of an aggregate showing a small total number of free parking places, it is not clear whether this is caused by a small number of parking places being free, or by a small number of free parking places having been *observed* in the time interval covered by the TTL, due to a generally low number of observations. Note that this is not a problem of our approach, but a general one. It is, fortunately, relatively easy to overcome. In the proposed application a car may distribute *two* values with separate sketches instead of just one: the number of observed free parking places, and in addition the number of observed total parking places. Both values in combination describe which *fraction* of the observed parking places is free. It generally seems that distributing such relative values is more robust. Recall that due to the soft-state approach it is not necessary to report occupied parking places since a no longer free parking place will die out if it is not refreshed.

This also accommodates trading off the considered timespan as introduced above: the application can easily infer the comprehensiveness and coverage of the underlying data basis for increasing time horizons, to optimize the tradeoff.

Sketches can be used to approximate sums of positive integers, but can be generalized to general integers and fixed or floating point numbers [22]. Our scheme is thus applicable whenever the aggregated value can be expressed through sums. Examples are counts, sums, or averages, but also variance and standard deviation (through the average and the average of the squares) or even products (by adding logarithms) [22]. The accuracies of the approximations, of course, vary, and an appropriate tradeoff for the specific application has to be found.

Further application examples could be the dissemination of the current traffic density (e. g., by distributing the number of observed vehicles and the total length of the roads for which there are observations), or the current average speed on a road. Both are useful to support navigation and route planning.

5. Extensions to the Basic Soft-State Sketch Algorithm

5.1. Compressing soft-state sketches

The techniques discussed above allow to summarize individual observations into hierarchical aggregates of increasing geographical scope. This reduces the number of data items that need to be exchanged in the network. However, the size of the individual soft-state sketches themselves is relatively large: each individual aggregate consists of $m \cdot w$ TTL counters. So, sketches in their standard representation consume significant network bandwidth when they are transmitted. This raises the question whether the size of transmitted sketches and thereby the bandwidth requirements can be reduced.

A close look easily reveals that sketches indeed carry a lot of redundancy. Positions on the left hand side of the sketch are much more likely to be "hit" when an item is added. In soft-state sketches, these entries are thus likely to have a high remaining TTL, i. e., a high value. On the right hand side of the sketch, in contrast, additions occur comparatively rarely, and it may be expected that many entries are zero. This redundant structure indicates that effective data compression is possible.

The local storage size of the knowledge base is much less critical than the network capacity constraints. Thus, soft-state sketches need not be compressed while being stored locally, facilitating an easy implementation of the update, merge and decay operations discussed above. A well-suited compression mechanism will therefore allow for efficient on-the-fly compression and decompression of sketches upon sending or receiving beacons, without causing too much computational overhead.

We will now introduce a lossless compression scheme for soft-state sketches. That the compression is lossless implies that the receiver can reconstruct the transmitted soft-state sketch exactly as it was. Thus, applying this compression scheme reduces the size of the transmitted sketch, but it does not affect the accuracy of the aggregates in any way.

In [26], a compression scheme for standard FM sketches with the desired characteristics has been proposed. The idea is to transmit the total number of leading ones in a PCSA set first. This value characterizes the probability distribution of the possible values (0 and 1) at each bit position. Both sender and receiver use this as a data model for arithmetic coding [29], thereby achieving compression very close to the entropy limit in an algorithmically very efficient way. Unfortunately, this idea cannot directly be applied to soft-state sketches, because individual entries in soft-state sketches can take more than two values, and the probability distributions cannot be parametrized as compactly as in the case of standard FM sketches.

Nevertheless, we may take up the general idea of modelling the interdependencies of the sketch entries' probability distributions, thereby exploiting our knowledge about the structure of the soft-state sketches for efficient compression. The task of a data model for arithmetic coding is to make "predictions" about the probabilities of all possible values for the next input character, given the input so far. The model is used by both encoder and decoder, so that they agree on the same probabilities. For good results, the quality of the model is crucial: better prediction results in better compression. So, given that we are able to formulate a model that makes good predictions about the values of the individual soft-state sketch entries, arithmetic coding constitutes an ideal basis for a tailored compression scheme.

In order to come up with such a model, we examine the structure of a PCSA set of soft-state sketches with maximum TTL T more closely. For all $t, 1 \leq t \leq T$, let x_t be the (unknown) number of distinct elements that have been added to the PCSA set, for which the remaining TTL is t (i. e., x_t distinct elements that have been added $T-t$ intervals ago). Let

$$X_t := \sum_{i=t}^{T} x_i. \tag{9}$$

In a PCSA set with m component sketches, the probability that the j-th entry in the i-th sketch, here denoted by $s_{i,j}$, is "hit" by an insertion is $2^{-j}/m$. Hence, the probability that, after X_t insertions of elements with TTL $\geq t$, position $s_{i,j}$ still has a TTL below t is

$$p_{j,t} := P(s_{i,j} < t) = \left(1 - \frac{2^{-j}}{m}\right)^{X_t}. \tag{10}$$

Note that $p_{j,t}$ does not depend on i, which is clear from the fact that there is no structural difference between the individual sketches in a PCSA set. That is, the probability of a

certain value does not depend on the index of the sketch in the PCSA set, but only on the position within the sketch.

For the entry one step to the left from $s_{i,j}$, that is, for $s_{i,j-1}$, the respective probability is

$$p_{j-1,t} = \left(1 - 2 \cdot \frac{2^{-j}}{m}\right)^{X_t}. \tag{11}$$

By solving (10) for X_t and then using the result in (11) we obtain

$$\begin{aligned} p_{j-1,t} &= \left(1 - 2 \cdot \frac{2^{-j}}{m}\right)^{\frac{\ln(p_{j,t})}{\ln(1-(2^{-j}/m))}} \\ &= \exp\left(\ln\left(\left(1 - 2 \cdot \frac{2^{-j}}{m}\right)^{\frac{\ln(p_{j,t})}{\ln(1-(2^{-j}/m))}}\right)\right) \\ &= \exp\left(\ln\left(1 - 2 \cdot \frac{2^{-j}}{m}\right) \cdot \frac{\ln(p_{j,t})}{\ln(1-(2^{-j}/m))}\right) \\ &= \exp\left(\ln\left((p_{j,t})^{\frac{\ln(1-2\cdot(2^{-j}/m))}{\ln(1-(2^{-j}/m))}}\right)\right) \\ &= (p_{j,t})^{\frac{\ln(1-2\cdot(2^{-j}/m))}{\ln(1-(2^{-j}/m))}}. \end{aligned} \tag{12}$$

Thus, based on the probability that the sketch entry $s_{i,j}$ has a value of less than t, we can now calculate the probability that entry $s_{i,j-1}$ is less than t. Therefore, we know how the probability distributions for the sketch entries in neighboring positions are interrelated. However, this does of course *not* tell us what the actual probabilities $p_{j,t}$ are. They depend on the individual sketch being processed.

Witten et al. [30] use arithmetic coding with a dynamic model. In their scheme, the input distribution is "learned" while processing the input. Basically, the algorithm keeps track of the input distribution in the data processed so far, continuously adjusting the model. However, this approach assumes that the input distribution is the same for all input characters—an assumption that does not hold when compressing soft-state sketches.

In our sketch compression scheme, we combine the idea of learning the input distribution on-the-fly with our knowledge about the interdependencies between the distributions at different positions in the sketch. We may think of a PCSA set as a matrix, as in Figure 4, where an example consisting of three sketches S_1, S_2, S_3 with six TTL counter positions each is shown. In our algorithm, we traverse the PCSA set "column-wise" from right to left, i. e., we start with the rightmost entries of all component sketches. Because these entries all exhibit the same probability distribution ($p_{j,t}$ does not depend on i), we may apply Witten at al.'s algorithm within this column of the matrix. While processing the entries, the algorithm counts the number of occurrences of each value and builds a continuously refined model for this column.

For instance, in the rightmost column in Figure 4—the first one to be processed—the algorithm will thus, during compression, learn that the value 0 occurred three times.

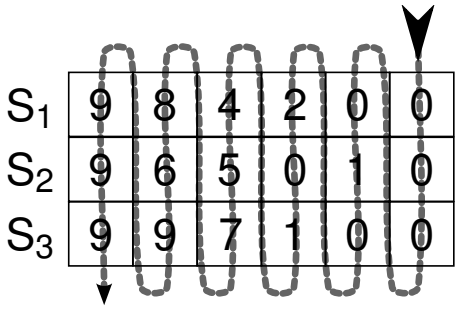


Figure 4: A PCSA set consisting of three soft-state sketches and how it is traversed for compression.

However, this does not immediately apply to other colums, where value 0 will not occur with the same probability. Generally, the further left a column is, the more likely higher values become. Before proceeding with the next column, we therefore have to make use of the interrelation of the probability distributions expressed in (12). Based on the probability distribution estimated by Witten et al.'s algorithm for column j, we can calculate an estimate for the probability distribution in column $j-1$. We then in turn use this transformed distribution as a starting point for the compression of column $j-1$. While the entries in this column are compressed, the model is again continuously refined and adjusted to the actual values, before it is again transformed according to (12) when transitioning to column $j-2$, and so on.

However, while (12) is an exact representation of the interdependencies between the probability distributions, it is not very well-suited for practical purposes, because its evaluation requires complex floating-point operations. We observe that for $m \to \infty$, (12) converges to

$$\lim_{m\to\infty} p_{j-1,t} = p_{j,t}^2. \tag{13}$$

This can be seen by applying the rule of Bernoulli-L'Hospital for $m \to \infty$ to the exponent. Some simple numerical experiments show that the convergence is very quick: for $m \geq 8$, $j \geq 1$ already, the difference is at most in the order of 0.01, and quickly decreases further for higher m and j. Such small differences do not result in any noticeable loss in compression performance. One may thus employ the much simpler formula

$$p_{j-1,t} = p_{j,t}^2 \tag{14}$$

in practical implementations as a very good approximation.

If (14) is used for transforming the estimated probabilities upon column transitions in combination with Witten et al.'s arithmetic coding algorithm, all steps necessary for compression and decompression can be performed with pure integer arithmetics in linear time. It is therefore ideally suited even for resource-constrained devices.

5.2. Longer counters for larger aggregates

In a typical application of our aggregation scheme, large aggregates will be distributed over longer distances, while smaller aggregates remain in the closer vicinity.

But distribution over long distances implies that the aggregates are longer underway, and will typically traverse more hops. In short: the information will be older when it arrives at the place where it is used. Consequently, while locally relatively quick ageing of the information will be tolerable (or even desirable), for larger, widely distributed aggregates it will likely be advisable to use a longer lifetime for the soft-state information. To accomplish this, it is possible to extend our algorithm in a way that uses soft-state sketches with longer TTL counters for larger aggregates. Obviously, this increases the size of these aggregates, but since the available TTL range grows exponentially with the counter size it scales very well.

Using different counter lengths in different aggregates slightly increases the complexity of the merge operation. However, the necessary modifications become relatively straightforward if the *age* of an entry is considered instead of its TTL. The age is the number of decrements that have occurred at that position, i. e., its difference from the maximum TTL. Instead of setting the position in the merged aggregate to the maximum TTL of the two merged soft-state sketches, it is set to the TTL corresponding to the *minimum age*, where the received TTL is larger than zero. The resulting operation is equivalent in the case of identical counter sizes in the two sketches, but differs if the counter sizes are different.

As an example, consider a locally stored aggregate with a counter size of eight bits. Let us focus on one single position, and assume that it currently has a value of 8. This corresponds to an age of $255-8=247$, since the maximum TTL is $2^8-1=255$ here. Now a sketch for a sub-area contained in our aggregate is received, which uses a counter size of only four bits. The value at our bit position is 10 in this sketch. Because the maximum TTL is 15, this corresponds to an age of 5. Since the minimum age of local and received aggregate is 5, we set the local aggregate to this age, and consequently to a TTL value of $255-5=250$.

More formally speaking, if T_{local} and s_{local} are the maximum TTL and the current value of an entry of the locally stored sketch, and T_{recv} and $s_{\text{recv}}>0$ are the corresponding counterparts in a received sketch that is to be merged in, then the new value of the respective position is given by

$$T_{\text{local}}-\min\{T_{\text{local}}-s_{\text{local}},\ T_{\text{recv}}-s_{\text{recv}}\}=\max\{s_{\text{local}},\ s_{\text{recv}}+T_{\text{local}}-T_{\text{recv}}\}. \qquad (15)$$

This scheme is applicable for arbitrary combinations of counter sizes.

6. Evaluation

6.1. Methodology

To evaluate our scheme, we implemented it in a simulation environment [31] encompassing the network simulator ns-2 [32] and the microscopic traffic simulator VISSIM [33]. VISSIM was used to generate the vehicular movements. We used a detailed model of a real city's extended downtown area with more than 500 km of roads and up to 10 000 vehicles. It includes, for example, multi-lane traffic, traffic lights, and different types of vehicles, and simulates differing driver psychologies. The vehicular traffic in the model is based on extensive measurements undertaken by the city administration for the purpose of traffic planning, and may therefore be expected to come close

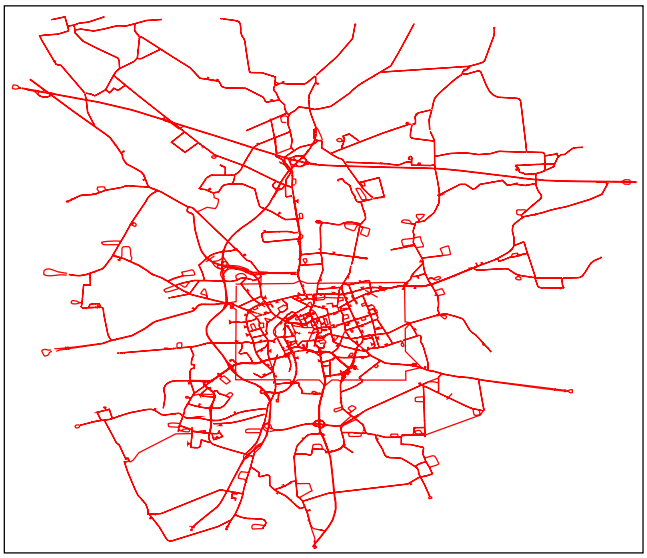


Figure 5: Road network used for the simulation.

to vehicular traffic patterns in real cities. The road network used in the simulation is depicted in Figure 5.

We simulate a VANET equipment penetration ratio of 2 %, 10 %, 20 % and 30 %. Unless otherwise stated, 20 % is used as a standard value in this section. In ns-2, IEEE 802.11 is employed as the MAC protocol, with the two-ray ground propagation model with a communication range of 250 meters and a carrier sense range of 550 meters. The network simulator is enhanced with an obstacle model that does not allow radio signals to propagate through the walls of buildings.

Because, in a model of the given size, a combined simulation of all aspects in parallel is extremely time-consuming and costly in terms of computation, we break it down to a three-step process. First, vehicle movements are generated using VISSIM. Subsequently, the beaconing process is simulated by ns-2. In this step, all VANET-equipped cars periodically send beacons of size 1096 byte (1024 byte payload plus headers) once every fixed beacon interval. We varied this interval between 2 and 10 seconds. For most results shown here we used a beacon interval of 5 seconds. The PHY and MAC models in ns-2 decide which of the beacons are received by which subset of the cars—note that this is not affected by the actual data contained in the beacons. Finally, the application logic is simulated, including the information exchanged in the beacons and the knowledge base of each car before and after each beacon transmission.

The road density, movement speed and pattern, etc. are, like in a real city, very heterogeneous in the simulation model. This makes the model realistic. But evaluating a protocol in such a complex environment in absolute terms is difficult. For example, if a high-speed road with lots of traffic connects two points that are 5 km apart, then one can expect well-working dissemination over that distance. If two other points with the same distance are separated by a municipal park, then the performance is bound to be

worse. Often, it can make a large difference whether a single vehicle carrying a piece of information gets through to a certain point and distributes its knowledge there.

In order to overcome these difficulties at least partially, we opted to use an optimal reference dissemination scheme in an otherwise exactly identical simulation setting for comparison. In this scheme, information is exchanged when a beacon is received. But the optimal reference scheme is not implemented as a "protocol" in the strict sense. In particular, it does not care about practical bandwidth limitations. In a simulator, the transmitted packet and the information actually exchanged between the (simulated) node objects are independent. The optimal reference exploits this fact and does *not* aggregate information at all. Instead, upon reception the receiver's knowledge base is updated to include all the knowledge of the sender. Since each measurement is annotated with a timestamp, optimal merging of knowledge bases is trivial. In order to avoid using too old measurements in the optimal reference evaluation, we use a timeout equivalent to the initial TTL of the soft-state sketches' entries.

Implementing this optimal reference scheme is easily possible in a simulator—by a direct exchange between the simulator objects of the communicating cars. Obviously, it is not possible to implement this in practice. However, it is well suited as a benchmark: with a practical protocol based on beacons and a knowledge base, the cars can never have better information.

We evaluate both our scheme and the optimal reference with an idealized application. We subdivide the city area using a grid with 256 small squares. We simulate a simple stochastic process for each of these squares, the current value of which can be "measured" by every car entering the respective square. This could be interpreted as "counting" the number of free parking places in this square, to stick with the example used above. It may likewise be considered an abstract model for any other time-varying, measureable quantity. In a real world application, one would most likely choose road segements or city areas instead of squares. We use sqares instead of application specific areas only because we are interested in the general performance of our aggregation scheme rather than that of one specific application.

Our simulations cover a timespan of 15 minutes. Aggregation generally makes sense only if the dynamics of areas which are geographically close together are not completely unrelated (which can of course be expected in the real world). Thus, the stochastic processes of the single locations are also not totally independent. All of them typically start relatively low, and increase substantially over the first half of the simulation, before they subsequently tend to decrease again. In relation to the relatively short total time, the changes are very rapid. They are thus very challenging for a dissemination protocol that needs to keep track of them. Since, as discussed above, disseminating relative values is much more appropriate than absolute sums, we also have one reference value per area—e. g., the "total number of parking places"—, which is likewise "observed" and distributed.

For hierarchical aggregation, groups of four neighboring areas are combined into a total of 64 medium-sized areas, and these are in the same way aggregated further to form 16 large-scale areas. For deciding which information is to be transmitted in a beacon, we constrain ourselves to a simple selection strategy—mainly to limit the overall complexity of the system and to allow for reasonable result interpretation. In our

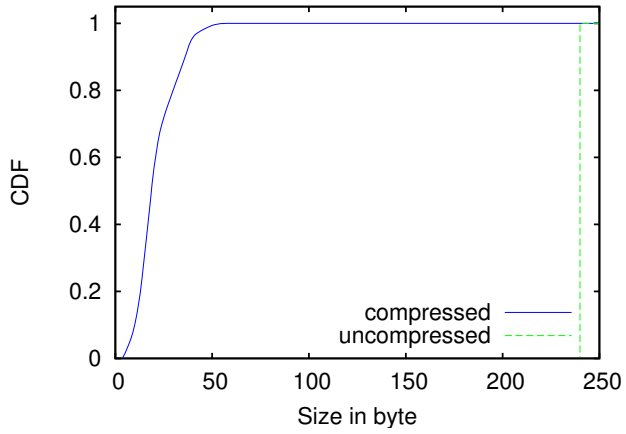


Figure 6: CDF of compressed sketch sizes.

simulations, each node transmits 27 aggregates in each beacon: on all three hierarchy levels, the area the car is currently in and the eight neighboring areas are sent.

6.2. Sketch size

Before we look at the information *within* the aggregates, let us first assess the performance of the compression scheme discussed in Section 5.1. In Figure 6, we show the cumulative distribution function of the compressed sizes of the aggregates transmitted in our simulation scenario. The dashed line shows the uncompressed size of the sketches, which is, of course, constant. We use PCSA sets of $m = 16$ sketches with length $w = 24$ and $n = 5$ bits per entry, i. e., the maximum TTL is $T = 31$. Therefore, the uncompressed size of an aggregate is 240 byte. From the figure, it is obvious that the compression yields a very significant size reduction: most sketches are compressed down to a size of 10–50 bytes. This confirms that the simulated beacon size of 1096 byte easily suffices to contain the 27 transmitted sketches.

Recall that the compression scheme introduced in Section 5.1 is lossless. Therefore, it has no influence on the accuracy of the aggregates.

6.3. Local accuracy

We now evaluate how well the aggregates reproduce the current value of a measured parameter *locally*, i. e., at the location where the measurement is performed. This is of interest because the data representation with soft-state sketches is probabilistic, and therefore does not necessarily reproduce the current value exactly.

Figure 7 shows, for one typical location, the average value from the knowledge bases of all vehicles that are currently within the corresponding region. As discussed above, we use a relative value, given by the ratio of the number currently free "parking places", which is changing over time, and their (fixed) total number for this location. For comparison, we also plot the current true value, and we perform the same evaluation with the optimal reference. In the specific case of a single, locally continuously

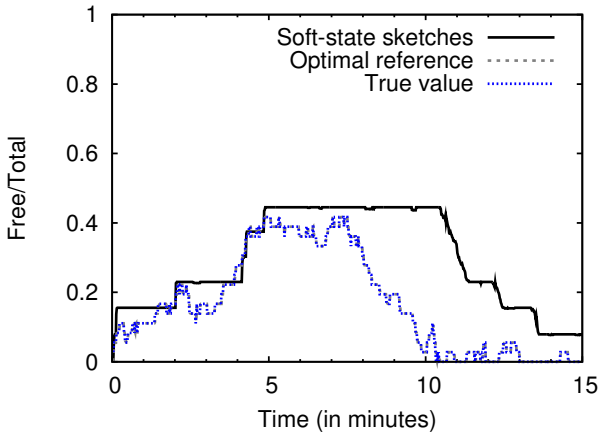


Figure 7: Accuracy of local sketches.

measurable value the latter two are of course always identical. Therefore, in the figure, the lines for the optimal reference and the true value are exactly on top of each other.

The figure shows that the sketch-based dissemination is able to model the correct value quite well. When the measured value starts decreasing again, the time needed for the soft-state decaying of the no longer set bits becomes visible: the sketch represents the maximum value observed in the recent past. Thus, there is a time lag before it follows a decreasing value. Due to the beaconing frequency of one beacon every five seconds and the maximum TTL of 31 beaconing cycles it is about $31 \cdot 5\,\mathrm{s} \approx 2.5\,\mathrm{min}$. Recall in this context that an application can dynamically select the cutoff horizon in (6), and may therefore tune this parameter locally and individually at any time.

Taking the time lag into account, the soft-state sketches indeed reflect the true situation very well here. This encouraging first result leads us to the next question: how well can aggregates for larger regions be formed and maintained?

6.4. Forming aggregates

With the proposed scheme, aggregates for larger regions can and will be formed wherever information on the respective region flows together. But nevertheless it is reasonable to expect that this will most regularly happen within the respective region. Therefore, we now look at how well sketch-based aggregates represent the situation in a larger area, while they are stored, passed around, and merged by cars within this area. Note that now the cars can no longer observe the entirety of the underlying information themselves. They can only measure the current value of their own location, and thus depend on received and merged information from other nodes in order to complete their picture.

In Figures 8 and 9 we show such evaluations for a typical medium-sized and a large inner-city aggregate (consisting of 4 and 16 locations, respectively). It is visible that perfect knowledge like in Figure 7 is no longer possible: the optimal reference does not always have perfect information. Especially for the high-level aggregate in Figure 9 the

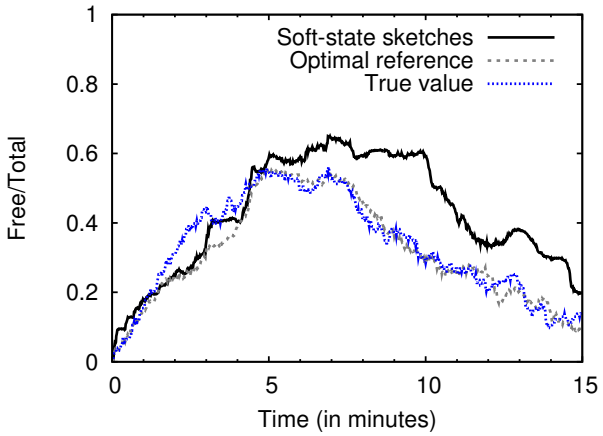


Figure 8: Accuracy of local medium-sized aggregate.

inevitable delay until up-to-date information on the entirety of the area has arrived can clearly be seen from the offset between the true value and the optimal reference. The sketch-based aggregation is close to the optimal knowledge, again with the previously discussed soft-state time lag in case of a decreasing value. These results demonstrate that probabilistic aggregation is indeed able to collect the available information, yielding aggregates that represent the available knowledge.

6.5. Distributing aggregates

Now, we look at further away regions and evaluate the representation of the situation in the knowledge bases of participating cars. Figures 10 and 11 represent two instances of medium-sized regions, as they are seen at a distance of about 3 km beeline. In addition to the time lag for the information transport it is not astonishing that even the optimal reference scheme does not always have complete information. If up-to-date data only on parts of the total aggregation area is present and this data is not typical for the whole aggregate, effects like the overestimation around simulation minute nine in Figure 11 are the logical consequence. Nevertheless, the estimates once again reflect the true situation in the modeled regions well. This is also confirmed by a look at a large aggregate's representation of the outskirt area in our model, as it is depicted in Figure 12.

The impact of the penetration ratio on the quality of aggregates can be observed in Figure 13. We specifically included Figure 13(a) with the intention to show that there are penetration ratios where not even the optimal reference is able to distribute the information over significant distances. This is a fundamental constraint of car-to-car communication, not a limitation of the approach presented in this paper. Figures 13(b), 13(c) and 13(d) then show that whenever the density is sufficiently high to allow for communication the aggregates are very close to what is achievable in the optimal case.

Finally, we investigated the impact of the beacon rate on the performance of our aggregation scheme. Because the soft state counters are updated once per beaconing in-

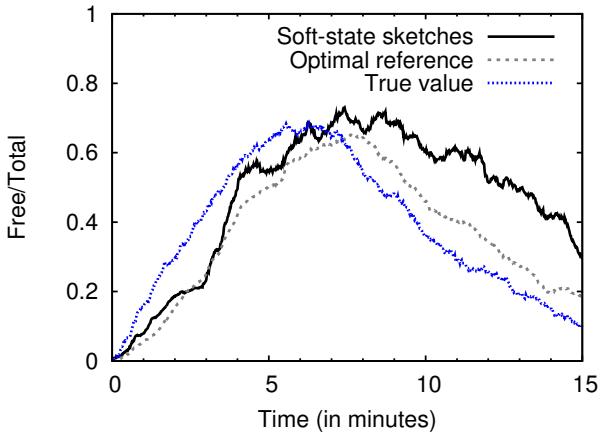


Figure 9: Accuracy of local large aggregate.

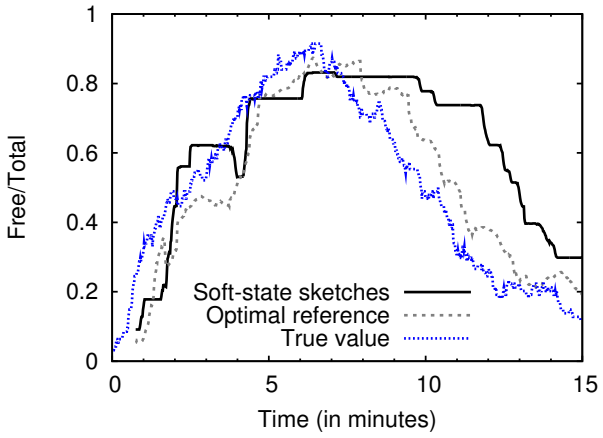


Figure 10: Accuracy of distant medium-sized aggregate.

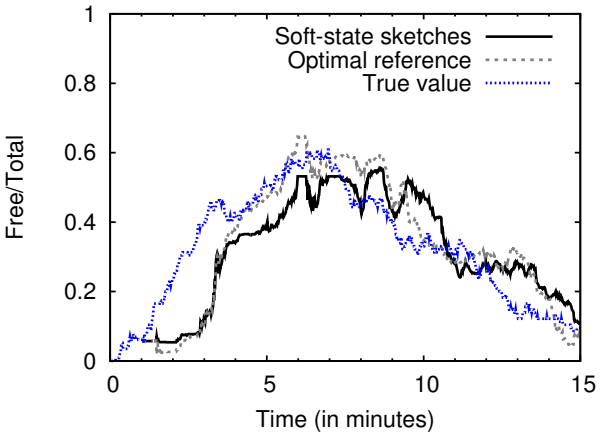


Figure 11: Accuracy of another distant medium-sized aggregate.

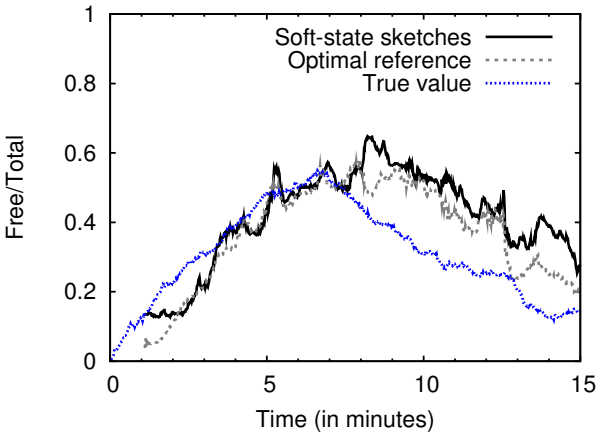
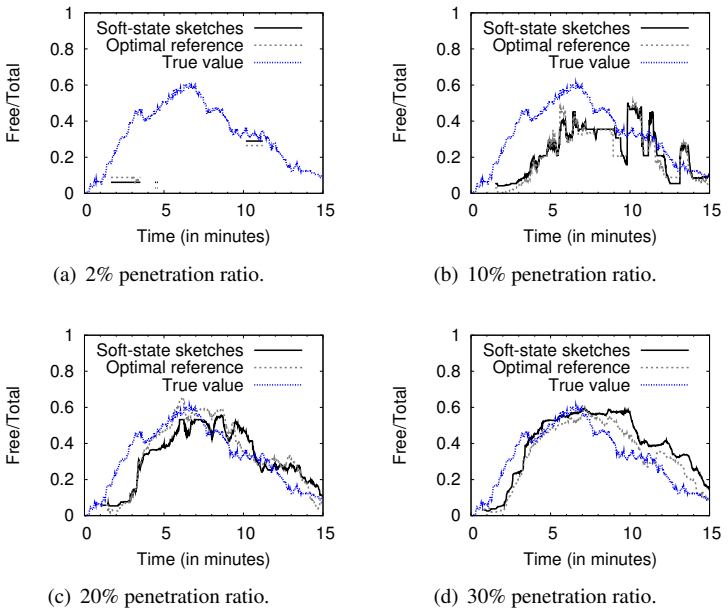


Figure 12: Accuracy of distant large aggregate.



(a) 2% penetration ratio.

(b) 10% penetration ratio.

(c) 20% penetration ratio.

(d) 30% penetration ratio.

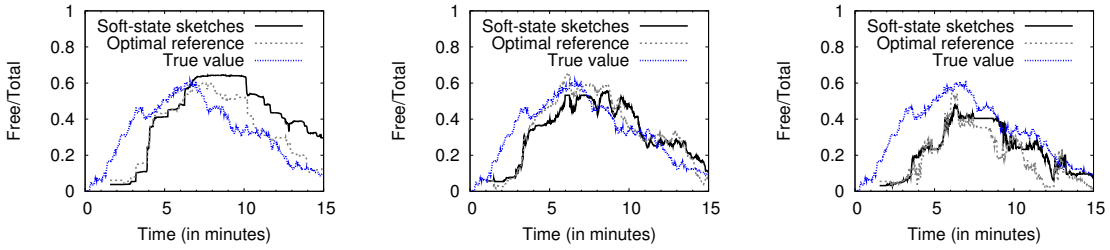
Figure 13: Impact of the pentration ratio on the aggregate from Fig. 11.

terval, we have adjusted the maximum counter values in these simulations accordingly, so that the soft state sketches keep an observation over identical time spans before it times out. Figure 14(a) (one beacon every two seconds) and Figure 14(b) (one beacon every five seconds) show that the approach is very stable over a wide range of beacon intervals. Only when the frequency of beacons becomes so low that communication opportunities are missed, as it is the case in Figure 14(c) (one beacon every ten seconds), the quality of the aggregated information starts to decline. Again, this is not a problem of our aggregation scheme but a general limitation of car-to-car communication, as the optimal reference shows exactly the same behaviour.

In summary, sketch-based probabilistic aggregation can be used to create aggregates that come close to what can theoretically be achieved with the considered kind of system. This optimum was here represented by a non-realizable optimal reference scheme. We consider these results very encouraging indications that the proposed algorithm is a suitable way to overcome the often observed general difficulties of distributed, uncoordinated data aggregation in dissemination schemes.

7. Conclusion

Data aggregation is a vital component of any scheme disseminating data in VANETs over significant distances. In this paper we proposed the first duplicate insensitive aggregation scheme for VANETs. It specifically supports hierarchical aggregation of information. The key idea of our approach is to extend Flajolet-Martin



(a) 2s beacon interval. (b) 5s beacon interval. (c) 10s beacon interval.

Figure 14: Impact of the beacon interval on the aggregate from Fig. 11.

sketches with the ability to time-out information through a soft state mechanism. We discussed several application scenarios of our scheme and possible extensions, showing its broad applicability. Furthermore we demonstrated how the soft-state sketches can be compressed efficiently, without loss, before being transmitted.

The performance of our aggregation scheme was evaluated in a simulation study based on ns-2 and a realistically modeled inner-city VANET. We compared the values to what could ideally be achieved in the same setting using a large variety of parameter settings such as penetration ratio and beacon interval. The results of this study confirm our expectation that sketch-based schemes are well-suited for the considered purpose, and may well form a central building block for a large variety of future VANET applications.

References

[1] T. Nadeem, S. Dashtinezhad, C. Liao, L. Iftode, TrafficView: traffic data dissemination using car-to-car communication, ACM SIGMOBILE Mobile Computing and Communications Review 8 (3) (2004) 6–19.

[2] L. Wischhof, A. Ebner, H. Rohling, M. Lott, R. Halfmann, SOTIS – a self-organizing traffic information system, in: VTC '03-Spring: Proceedings of the 57th IEEE Vehicular Technology Conference, 2003, pp. 2442–2446.

[3] L. Wischhof, A. Ebner, H. Rohling, Information dissemination in self-organizing intervehicle networks, IEEE Transactions on Intelligent Transportation Systems 6 (1) (2005) 90–101.

[4] M. Caliskan, D. Graupner, M. Mauve, Decentralized discovery of free parking places, in: VANET '06: Proceedings of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks, 2006, pp. 30–39.

[5] B. Scheuermann, C. Lochert, J. Rybicki, M. Mauve, A fundamental scalability criterion for data aggregation in VANETs, in: MobiCom '09: Proceedings of the 15th Annual ACM International Conference on Mobile Computing and Networking, 2009, pp. 285–296.

[6] J. Brønsted, L. M. Kristensen, Specification and performance evaluation of two zone dissemination protocols for vehicular ad-hoc networks, in: ANSS '06: Proceedings of the 39th Annual Simulation Symposium, 2006, p. 12. doi:10.1109/ANSS.2006.43.

[7] K. Ibrahim, M. C. Weigle, Accurate data aggregation for VANETs, in: VANET '07: Proceedings of the 4th ACM International Workshop on Vehicular Ad Hoc Networks, 2007, pp. 71–72, poster Session.

[8] K. Ibrahim, M. C. Weigle, Optimizing CASCADE data aggregation for VANETs, in: MoVeNet '08: Proceedings of the 2nd International Workshop on Mobile Vehicular Networks, 2008.

[9] C. Lochert, B. Scheuermann, M. Mauve, Probabilistic aggregation for data dissemination in VANETs, in: VANET '07: Proceedings of the 4th ACM International Workshop on Vehicular Ad Hoc Networks, 2007, pp. 1–8. doi:10.1145/1287748.1287750.

[10] P. Flajolet, G. N. Martin, Probabilistic counting algorithms for data base applications, Journal of Computer and System Sciences 31 (2) (1985) 182–209.

[11] H. Xu, M. Barth, An adaptive dissemination mechanism for intervehicle communication-based decentralized traffic information systems, in: ITSC '06: Proceedings of the 9th International IEEE Conference on Intelligent Transportation Systems, 2006, pp. 1207–1213.

[12] T. Fujiki, M. Kirimura, T. Umedu, T. Higashino, Efficient acquisition of local traffic information using inter-vehicle communication with queries, in: ITSC '07: Proceedings of the 10th International IEEE Conference on Intelligent Transportation Systems, 2007, pp. 241–246. doi:10.1109/ITSC.2007.4357748.

[13] L. Wischhof, A. Ebner, H. Rohling, M. Lott, R. Halfmann, Adaptive broadcast for travel and traffic information distribution based on inter-vehicle communication, in: IV '03: Proceedings of the IEEE Intelligent Vehicles Symposium, 2003, pp. 6–11. doi:10.1109/IVS.2003.1212873.

[14] H. Wu, R. Fujimoto, R. Guensler, M. Hunter, MDDV: A mobility-centric data dissemination algorithm for vehicular networks, in: VANET '04: Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks, 2004, pp. 47–56. doi:http://doi.acm.org/10.1145/1023875.1023884.

[15] C. Adler, R. Eigner, C. Schroth, M. Strassberger, Context-adaptive information dissemination in VANETs – maximizing the global benefit, in: CSN '06: Proceedings of the 5th IASTED International Conference on Communication Systems and Networks, 2006, pp. 7–12.

[16] P. Golle, D. Greene, J. Staddon, Detecting and correcting malicious data in VANETs, in: VANET '04: Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks, 2004, pp. 29–37. doi:http://doi.acm.org/10.1145/1023875.1023881.

[17] M. Jakobsson, S. Wetzel, Efficient attribute authentication with applications to ad hoc networks, in: VANET '04: Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks, 2004, pp. 38–46. doi:http://doi.acm.org/10.1145/1023875.1023882.

[18] M. Raya, A. Aziz, J.-P. Hubaux, Efficient secure aggregation in VANETs, in: VANET '06: Proceedings of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks, 2006, pp. 67–75.

[19] F. Picconi, N. Ravi, M. Gruteser, L. Iftode, Probabilistic validation of aggregated data in vehicular ad-hoc networks, in: VANET '06: Proceedings of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks, 2006, pp. 76–85.

[20] S. Park, C. C. Zou, Reliable traffic information propagation in vehicular ad-hoc networks, in: SARNOFF '08: Proceedings of the IEEE Sarnoff Symposium, 2008, pp. 1–6. doi:10.1109/SARNOF.2008.4520100.

[21] K. Ibrahim, M. C. Weigle, Securing CASCADE data aggregation for VANETs, in: MoVeNet '08: Proceedings of the 2nd International Workshop on Mobile Vehicular Networks, 2008, poster Session.

[22] J. Considine, F. Li, G. Kollios, J. Byers, Approximate aggregation techniques for sensor databases, in: ICDE '04: Proceedings of the 20th International Conference on Data Engineering, 2004, pp. 449–460.

[23] Y. Tao, G. Kollios, J. Considine, F. Li, D. Papadias, Spatio-temporal aggregation using sketches, in: ICDE '04: Proceedings of the 20th International Conference on Data Engineering, 2004, pp. 214–226.

[24] M. Durand, P. Flajolet, Loglog counting of large cardinalities, in: ESA '03: Proceedings of the 11th Annual European Symposium on Algorithms, 2003, pp. 605–617.

[25] P. Flajolet, É. Fusy, O. Gandouet, F. Meunier, Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm, in: AofA '07: Proceedings of the 2007 International Conference on Analysis of Algorithms, 2007.

[26] B. Scheuermann, M. Mauve, Near-Optimal Compression of Probabilistic Counting Sketches for Networking Applications, in: Dial M-POMC 2007: Proceedings of the 4th ACM SIGACT-SIGOPS International Workshop on Foundation of Mobile Computing, 2007.

[27] P. Kirschenhofer, H. Prodinger, W. Szpankowski, How to count quickly and accurately: A unified analysis of probabilistic counting and other related problems, in: ICALP '92: Proceedings of the 19th International Colloquium on Automata, Languages and Programming, 1992, pp. 211–222.

[28] M. Piorkowski, M. Grossglauser, A. Papaioannou, Mobile user navigation supported by WSAN: Full-fledge demo of the SmartPark system, in: MobiHoc '06: Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2006, Technical Demonstration.

[29] J. J. Rissanen, G. G. Langdon, Arithmetic coding, IBM Journal of Research and Development 23 (2) (1979) 149–162.

[30] I. H. Witten, R. M. Neal, J. G. Cleary, Arithmetic coding for data compression, Communications of the ACM 30 (6) (1987) 520–540.

[31] C. Lochert, M. Caliskan, B. Scheuermann, A. Barthels, A. Cervantes, M. Mauve, Multiple Simulator Interlinking Environment for Inter Vehicle Communication, in: VANET '05: Proceedings of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks, 2005, pp. 87–88. doi:10.1145/1080754.1080771.

[32] The network simulator ns-2, http://www.isi.edu/nsnam/ns/.

[33] PTV AG, The VISSIM traffic simulator, http://www.ptvag.com/software/transportation-planning-traffic-engineering/software-system-solutions/vissim/.